Towards User-Centric Metrics for Denial-Of-Service Measurement [*]

Jelena Mirkovic, University of Delaware
Alefiya Hussain, SPARTA, Inc.
Brett Wilson, SPARTA, Inc.
Sonia Fahmy, Purdue University

Peter Reiher, UCLA
Roshan Thomas, SPARTA, Inc.
Wei-Min Yao, Purdue University
Stephen Schwab, SPARTA, Inc.

ABSTRACT

To date, the measurement of user-perceived degradation of quality of service during denial of service (DoS) attacks remained an elusive goal. Current approaches mostly rely on lower level traffic measurements such as throughput, utilization, loss rate, and latency. They fail to monitor all traffic parameters that signal service degradation for diverse applications, and to map application quality-of-service (QoS) requirements into specific parameter thresholds. To objectively evaluate an attack's impact on network services, its severity and the effectiveness of a potential defense, we need precise, quantitative and comprehensive DoS impact metrics that are applicable to any test scenario.

We propose a series of DoS impact metrics that measure the QoS experienced by end users during an attack. The proposed metrics consider QoS requirements for a range of applications and map them into measurable traffic parameters with acceptable thresholds. Service quality is derived by comparing measured parameter values with corresponding thresholds, and aggregated into a series of appropriate DoS impact metrics. We illustrate the proposed metrics using extensive live experiments, with a wide range of background traffic and attack variants. We successfully demonstrate that our metrics capture the DoS impact more precisely than the measures used in the past.

Categories and Subject Descriptors: C.4 Performance of systems: Measurement techniques
General Terms: Measurement, security, standardization.
Keywords: Denial of service, metrics.

1. INTRODUCTION

Denial of service (DoS) is a major threat today. Its intended effect is to prevent legitimate users from doing routine business with the victim, by exhausting some limited resource via a packet flood or by sending malformed packets that cause network elements to crash. Service denial is experienced by users as a severe slowdown, service quality degradation or a complete disruption of communication with the victim.

DoS attacks are studied through network simulation or live experimentation. Accurately measuring DoS impact is essential for evaluation of potential DoS defenses and study of novel attack vectors. Current approaches to quantify the impact of DoS attacks involve a collection of one or several traffic measurements and a comparison of their first order statistics (e.g., mean, standard deviation, minimum or maximum) or the value distributions in the baseline and the attack case. If a defense is being evaluated, the statistics are also collected for the scenario with attack/defense combination. Frequently used traffic measurements include the legitimate traffic request/response delay, total time needed to complete a legitimate transaction, legitimate traffic goodput or throughput at the victim, legitimate traffic loss, and division of a critical resource between the legitimate and the attack traffic.

We argue that current DoS measurement approaches are *incomplete* and *imprecise*. Because there is no consensus within the community on which measurements best reflect the DoS impact, researchers frequently choose one they feel is the most relevant. This causes the results to be incomplete, as each independent traffic measurement captures only one aspect of the service denial. For example, a prolonged request/response time will properly signal denial of service for two-way applications, such as HTTP, FTP and DNS, but not for media traffic that is sensitive to one-way delay, packet loss and jitter. The lack of a common DoS impact measure prevents comparison among published work. We further argue that the current measurement approaches are imprecise, because ad-hoc comparisons of measurement statistics or distributions cannot accurately signal service denial. These can only express that network traffic behaves *differently* under attack, but do not accurately measure which services have been denied and how severely. We survey these existing DoS impact metrics in Section 2.

We propose a novel, user-centric approach to DoS impact measurement. Our key insight is that DoS always causes degradation of service quality, and a metric that holistically captures a user's QoS perception will be applicable to all test scenarios, regardless of the attack strategy and the legitimate traffic mix. For each popular network application, we specify its QoS requirements, which consist of relevant traffic measurements and corresponding thresholds that define good service range. Each legitimate transaction is then evaluated against the QoS requirements for its application category, and transactions that do not meet all the requirements are considered "failed." We aggregate the information about transaction failure into several simple and intuitive qualitative and quantitative composite metrics to expose the precise interaction of the DoS attack with the legitimate traffic: the impact of attack on vari-

[*]This material is based on research sponsored by the Department of Homeland Security under agreement number FA8750-05-2-0197. The views and conclusions contained herein are those of the authors only.

ous applications and its severity, and times when failures occur. We describe our proposed metrics in Section 3.

Ideally, we would like to validate our metrics against the "ground truth," e.g., using a set of traces that contain a real user's communication with a network under attack, and user feedback about the time and severity of observed service denial. We are currently conducting a study to collect such traces and user feedback, and we plan to report these results in our future work. In this paper, our experiments focus on reproducing a wide range of popular attacks that other researchers have studied and that are known to produce a certain level of service denial, in controlled testbed experiments and NS-2 simulations, but without human user participation. Section 4 describes our testing methodology. We demonstrate the inadequacy of the existing metrics and the utility of our proposed metrics in Section 5, via live experiments on the DETER testbed [4]. We illustrate the utility of our proposed metrics in NS-2 simulations in Section 6. We survey related work in Section 7 and conclude in Section 8.

The contribution of this paper is three-fold: (1) We propose a novel approach to DoS impact measurement that relies on easy-to-compute application-specific QoS requirements. Although our proposed metrics combine several existing metrics, our contribution lies in (i) the careful specification of which traffic measurements reflect service denial for a range of the most popular applications, (ii) the definition of QoS thresholds for each measurement and each application class, and (iii) the aggregation of multiple measurements, using the threshold-based model, into intuitive and informative DoS metrics that can be directly applied to traffic traces, without having to define complex application utility functions or instrument each individual application. (2) We present several intuitive metrics that comprehensively capture the DoS impact in a variety of test scenarios, in testbed experimentation or in simulation. (3) We demonstrate the utility of proposed metrics through rigorous experimentation on DETER testbed [4] under a range of DoS attacks. To our knowledge, there have been no previous attempts to consolidate state-of-the-art research to identify comprehensive, precise and widely applicable metrics for DoS impact evaluation.

We have made all the data used for this paper available at http://www.cis.udel.edu/∼sunshine/expcs/data.

2. LEGACY METRICS

Existing DoS research has focused on measuring denial of service through selected legitimate traffic parameters. The commonly used parameters are: (a) packet loss, (b) traffic throughput or goodput, (c) request-response delay, (d) duration of transaction, and (e) allocation of resources. Researchers have used both the simple metrics (single traffic parameter) and combinations of these to report the impact of an attack on the network. We now discuss each legacy metric and its failure in capturing the end-to-end DoS impact.

Loss is defined as the number of packets or bytes lost due to the interaction of the legitimate traffic with the attack [28]. As a metric of service denial, loss can be inferred from TCP traffic retransmissions, or measured directly as number of packets that did not reach their destination. As a metric of collateral damage, loss is reported as the number (or percentage) of legitimate packets discarded by the defense system.

The loss metric primarily expresses the presence and extent of congestion in the network due to flooding attacks. However, it is ill-suited for attacks that do not continually create congestion, such as the recently examined class of pulsing attacks [18, 14] that will create congestion periodically for short intervals. In this class, service is denied because of TCP's congestion response which leads to reduced sending rate and prolonged transaction duration. Further, loss metrics usually do not distinguish between the types of packets lost. Since some packet losses have a more profound impact than others (for example, a lost SYN packet compared to a lost data packet), this metric is incomplete.

Throughput is defined as the number of bytes transferred per unit time from the source to the destination. **Goodput** is similar to throughput but it does not count retransmitted bytes [18, 16].

Both throughput and goodput metrics are meaningful for TCP-based traffic, which responds to congestion by lowering its sending rate. Indirectly, these metrics capture the presence and extent of congestion in the network and the prolonged duration of legitimate transactions due to congestion. These metrics cannot be applied to applications that are sensitive to jitter or to loss of specific (e.g., control) packets, because a high throughput level may still not satisfy the quality of service required by the user. Further, the throughput and goodput metrics do not effectively capture DoS impact on traffic mixes consisting of short connections, with a few packets to be sent to the server. Such connections already have a low throughput so service denial may be masked.

Finally, the throughput and goodput depend both on the volume and timing of individual transactions, as well as on network conditions. Unless the tests are perfectly repeatable, it is difficult to quantitatively compare values between two test runs.

Request-response delay is defined as the time lapse between when a request is first sent and when a complete response is received from the destination [15]. It is well-suited to measure the service denial of interactive applications, where a human user interacts with a server and expects to receive a response within a short time. On the other hand, non-interactive applications (e.g., email) have very different thresholds for acceptable request-response delay, while one-way traffic, such as media traffic, does not generate responses but is sensitive to one-way delay, loss and jitter.

Transaction duration is defined as the time between the start and the end of a data transfer between a source and destination [30, 20, 27]. The main drawback of this metric is that it heavily depends on the volume of data being transferred and whether the involved application is interactive and congestion-sensitive. In case of one-way transactions, such as media streaming, that do not respond to congestion, transaction duration will not be affected by the attack.

Allocation of resources is defined as the fraction of a critical resource (usually bandwidth) allocated to legitimate traffic vs. attack traffic [20, 24]. This metric does not provide any insight into the user-perceived service quality, but assumes that the only damage to legitimate traffic is inflicted by the lack of resources, which is applicable only to flooding attacks. Further, it cannot capture the collateral damage of a given defense. For example, a defense that drops 90% of legitimate and 100% of attack traffic, would appear perfect, since it allocates all resources to legitimate traffic.

In summary, the existing metrics suffer from two major drawbacks: (1) They measure a single traffic parameter assuming that its degradation can be used as a reliable signal of service denial. This approach is faulty as traffic parameters that signal service denial are application-specific. Further, different attack strategies can deny service without affecting the monitored parameter. (2) They fail to define the parameter range that is needed for acceptable service quality. For example, while low throughput or high packet loss can signify service denial, there is a lack of understanding how large a degradation must be so that a user experiences poor service quality. Such thresholds are application and task specific.

Finally, the existing metrics predominantly capture the service denial at the network layer, en route to the victim server. While many attacks target this route, some affect the server host or the application directly, target supporting network services (such as

DNS), or the route from the server to legitimate users. Existing metrics fail to capture the impact of these attacks.

3. PROPOSED DOS IMPACT METRICS

We now introduce several definitions that will help us define DoS impact metrics. The client is the host that initiates communication with another party which we will call the server.

Definition 1: A ***conversation*** between a client and a server is the set of all packets exchanged between these two hosts with a goal to provide a specific service to the client, at a given time. A conversation is explicitly initiated by a client application (e.g., by opening a TCP connection or sending a UDP packet to a well-known service) and ends either explicitly (a TCP connection is closed, UDP service is rendered to the client) or after a very long period of inactivity. If several channels are necessary to render service to a client, such as FTP control and data channel, or media control and data channel, all related channels are part of a single conversation.

Definition 2: A ***transaction*** is the part of a conversation that represents a higher-level task whose completion is meaningful to a user. A transaction usually involves a single request-reply exchange between a client and a server, or several such exchanges that occur close in time.

Definition 3: A transaction is considered ***successful*** if it meets all the QoS requirements of its corresponding application. If at least one QoS requirement is not met, a transaction is considered ***failed***.

3.1 The *pft* Measure

Our main DoS impact measure is the percentage of failed transactions (*pft*) in each application category. This metric directly measures the impact of a DoS attack on network services quantifying the quality of service experienced by end users. We first identify a set of popular applications in today's Internet and the traffic measurements the values of which indicate if the particular application's service was denied.

We then interpret the traffic as series of transactions that represent higher-level tasks whose completion is meaningful to a user, such as browsing one Web page, downloading a file, or having a VoIP conversation. Our main motivation for introducing the notion of a transaction is to properly handle lengthy communications that may involve many shorter, self-contained tasks. For example, if a user downloads 100 files, one by one, during a single FTP session, and the last file transfer fails, this does not indicate poor service for the entire FTP session but for 1% of tasks contained in this session. Section 3.3 describes our approach for identifying transactions.

We define a threshold-based model for the relevant traffic measurements, which is application-specific. When a measurement exceeds its threshold, this indicates poor service quality. Our threshold specifications are guided by the past findings in the area of QoS research [5, 29, 3, 26] and efforts of large standard bodies to define QoS requirements for next generation telecommunication networks. Specifically, we leveraged much of the 3rd Generation Partnership Project (3GPP) Specification of QoS Requirements [21]. The 3GPP [1] is a consortium of large telecommunications standards bodies from all over the world, that aims to "produce globally applicable Technical Specifications ... for a 3rd Generation Mobile System." Thus, the proposed set of QoS requirements has the advantage of being, to a large extent, ratified by the world's largest standards bodies.

We evaluate transaction success or failure using definition 3, and produce the *pft* measure for each application. We further aggregate the *pft* measure into several metrics, as we describe in Section 3.4.

3.2 Application QoS Requirements

Several organizations that collect and publish traffic traces [11, 25] analyze Internet applications and the ratio of the packets and bytes that they contribute to these traces. We surveyed their findings in order to assemble a list of popular applications. We further observe that the work of the 3GPP consortium has tackled the problem of defining popular applications and their QoS requirements in great depth [21], and we leverage their findings to extend and refine our compendium. Table 1 summarizes the application categories we propose, and their corresponding QoS requirements. The remainder of this section provides the rationale for our measurement and threshold selections. We note that should novel applications become popular in the future, the proposed application categories will need to be extended, but our DoS impact metrics will be immediately applicable to new applications.

Interactive applications such as Web, file transfer, telnet, email (between a user and a server), DNS and Ping involve a human user requesting a service from a remote server, and waiting for a response. For such applications, the primary QoS requirement is that a response is served within a user-acceptable delay. Research on human perception of Web traffic delay has shown that people can tolerate higher latencies for entire task completion if some data is served incrementally [5]. We specify two types of delay requirements for email, Web, telnet and file transfer transactions where a user can utilize a partial response: (a) *partial delay* measured between receipt of any two data packets from the server. For the first data packet, partial delay is measured from the end of a user's request, and (b) *whole delay* measured from the end of a user's request until the entire response has been received. Additionally, Telnet serves two types of responses to a user: it echoes characters that a user types, and generates a response to a user's request. The echo generation must be faster than the rest of response so we define the *echo delay* requirement for Telnet transactions. We identify the echo delay as the delay between a user's request and the first response packet.

We use 250 ms as the Telnet's echo delay requirement, as suggested in [21]. We use 4 s as the partial-delay threshold for Web, Telnet and email applications [21], and 10 s for file transfer applications [21]. We use 60 s as the whole-delay requirement for Web [5], and require that the delay for email and file transfer not exceed three times the expected delay [10], given the amount of data being transferred. The expected delay is defined as the delay experienced by the same transaction in the absence of an attack. For DNS and Ping services, we adopt a 4 s whole delay requirement. This is the maximum human-acceptable delay for interactive tasks [21]. We regard peer to peer applications as file transfer.

Media applications such as conversational and streaming audio and video have strict requirements for low loss, low jitter and low one-way delay. These applications further involve the media channel (where the audio and video traffic are sent, usually via UDP) and the control channel (for media control). Both of these channels must provide satisfactory service to the user. We adopt the one-way delay and loss requirements for media traffic from [21]. Because many media applications can sustain higher jitter than 1 ms [21] using variable-size buffers, we adopt the jitter threshold value of 50 ms as defined in [2]. Further, we treat the control traffic as interactive traffic and impose on it a 4 s partial-delay requirement.

Online games have strict requirements for low one-way delay [21]. We differentiate between first-person shooter (FPS) and real time strategy (RTS) games, because research has shown that their QoS requirements differ. We use the work presented in [3] (FPS) and [26] (RTS) as sources for specifying delay and loss bounds (see Table 1 for specific values).

Chat applications can be used for text and media transfer between two human users. While the request-response delays depend on human conversation dynamics, the receipt of user messages by the server must be acknowledged within a certain time. We express this delay requirement as 4 s threshold on a round-trip time between the client and the server. Additionally, we apply the QoS requirements for media applications to the media channel of the chat application.

Non-interactive services such as email transfer between servers and Usenet do not have a strict delay requirement. Instead, users are prepared to endure long delays, as long as the transactions complete within a given interval. 3GPP [21] specifies the transaction duration threshold as *several hours* for email and Usenet. We quantify this as 4 hours, since this value is commonly used by mail servers to notify a user about a failure to deliver a mail to the destination server.

Category	One-way delay	Req/resp delay	Loss	Dur.	Jitter
email (srv/srv)		whole, RTT < 4 h			
Usenet		whole, RTT < 4 h			
Chat, typing		RTT < 4 s			
Chat, audio	< 150 ms	whole, RTT < 4 s	$< 3\%$		< 50 ms
Chat, video	< 150 ms	whole, RTT < 4 s	$< 3\%$		
Web		part, RTT < 4 s		< 60 s	
FTP Data		part, RTT < 10 s		$< 300\%$	
FTP Control		part, RTT < 4 s			
FPS games	< 150 ms		< 3 %		
RTS games	< 500 ms				
Telnet		part, RTT < 250 ms			
email (usr/srv)		part, RTT < 4 s		$< 300\%$	
DNS		whole < 4 s			
Ping		whole < 4 s			
	media	control	media		media
Audio, conv.	< 150 ms	whole, RTT < 4 s	$< 3\%$		< 50 ms
Audio, messg.	< 2 s	whole, RTT < 4 s	$< 3\%$		< 50 ms
Audio, stream	< 10 s	whole, RTT < 4 s	$< 1\%$		< 50 ms
Videophone	< 150 ms	whole, RTT < 4 s	$< 3\%$		
Video, stream	< 10 s	whole, RTT < 4 s	$< 1\%$		

Table 1: Application categories and their QoS requirements

3.3 Measurement Approach

When devising a measurement methodology, it is important to ensure that the measurement does not perturb the system. We explored two possible approaches to collect the necessary measurements during experimentation: (i) we can instrument each client application to compute statistics such as average response time and transaction duration, or (ii) we can use real, uninstrumented application programs, and then collect and process network traffic traces to identify transactions, collect required traffic measurements and compute performance metrics in an automated fashion. The first approach (instrumented client) has the advantage that it can precisely identify transactions, because we have complete access to the application and transaction semantics. The downside is that we would need to instrument each client of interest, which would certainly limit experimentation to a chosen set of open-source clients. Our goal, however, is to devise a general measurement approach that is easily applicable to most test scenarios. The trace-based approach has exactly this advantage — all traffic can be collected by tcpdump, regardless of the application that generated it, and be subject to analysis, with no application modification. Thus, the trace-based approach scales better to new, off-the-shelf, diverse application types, and allows researchers to evaluate performance in traces captured by others. The disadvantage of the trace-based approach is that it can only observe anomalies at the network layer. If a user's host or a user's client application fails to send or display traffic to the user, or if a server returns a bogus reply, these events are not visible at the network layer.

In implementing trace-based service quality evaluation, we have encountered additional challenges that have led us to further refine our transaction success computation methods as follows: (1) We measure request/response delay and transaction duration using a sender-collected trace, which allows us to capture a user's QoS experience for a variety of DoS attacks, regardless of the resource they target. We correlate sender/receiver traces to measure one-way delay, loss and jitter. (2) We capture request/response delay at the transport level of the flow traffic. This measure will miss the delay that occurs if some request packets are dropped and retransmitted, but this delay is noticed by a user and must be included in success calculation. We thus add the maximum Round-Trip-Time (RTT) as an additional QoS requirement for TCP-based applications. (3) Although reference [21] specifies some loss bounds for TCP-based applications, we ignore these loss bounds because losses will either be handled through TCP retransmissions or will lead to a high request/response delay or RTT that exceeds the corresponding threshold. (4) If a DNS or a Ping request or reply is lost, we map this packet loss into delay, by setting request/response delay to a large, fixed value (10 times the delay threshold). (5) We measure one-way delay by matching packets from a sender's trace to packets in a receiver's trace. Successful matches update our delay estimate, but lost packets do not. (6) We differentiate between the total packet loss measured over the entire transaction duration, and the interval packet loss, measured over the most recent 5-second interval. A lengthy transaction that had excessive loss near the end of its life will have a low total loss measure but a high interval loss measure. We calculate the maximum interval loss during the experiment and use this value for success calculation. (7) If FTP transfer is performed with an FTP server, there will be a control and a data channel. Similar to our success criteria for media traffic, we impose a 4 s whole delay requirement on FTP control traffic, and we pair the data and control channels into a single transaction. If either channel fails, the entire transaction is considered as failed.

Within a trace, we identify client-initiated conversations using the following criteria: (1) For Web, email, Usenet, FTP control, Chat and Telnet traffic, we look for SYN packets sent from the client and use them to signify the start of a conversation. The conversation ends after the exchange of a FIN/ACK packet in each direction, or after any side sends a RST packet. If there is a denial of service, the conversation may also end when the service is denied, i.e., when one of the QoS thresholds has been exceeded. (2) For FTP data traffic, we look for SYN packets sent to this client and associate the FTP data channel with the corresponding FTP control channel. This association is done by parsing the content of FTP control packets and extracting data port information, then locating a conversation with the corresponding port at the server side and port 20 at the client side. (3) For DNS traffic, packets sent to port 53 with a request bit set will initiate a new conversation if they are not retransmissions. We identify retransmitted DNS requests using the identifier in the DNS header. (4) ICMP ECHO packets with a request bit set will initiate a new conversation if they are not retransmissions. We identify retransmitted ICMP ECHO requests using the identifier in the ICMP header. (5) For VoIP traffic, we currently use a VoIP simulator that generates a constant-rate UDP packet stream. Hence, we identify new conversations by looking for packets sent to the assigned VoIP server port.

Within client-server conversations, we identify *transactions* as user/server exchanges that are meaningful to the user. As mentioned above, had we instrumented applications, we could have precisely identified transaction semantics. Instead, traffic trace analysis is imperfect in transaction identification, but is more extensible and portable.

Table 2 shows how we identify transactions in the trace data. For

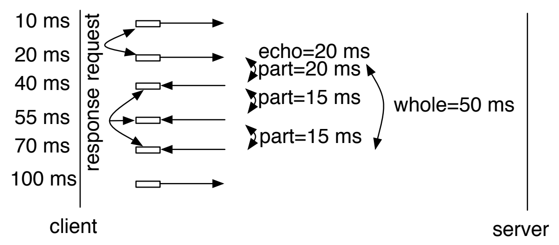


Figure 1: Illustration of request/response identification

interactive applications, an inactive time (user think time) followed by a new user's request denotes a new transaction. A transaction is either a part of a flow, or the entire flow, e.g., if a user opened a TCP connection to an FTP server, downloaded one file and closed the connection, this would be recognized as one transaction. Downloading 3 files in the same session, with think time in-between, would be recognized as 3 transactions. In case of media traffic, both the media stream (UDP flow) and the control stream (TCP flow) are part of a single transaction. Similarly, for FTP transactions both control and data channels are part of a single transaction.

Application	Transaction
email (srv/srv), Usenet	TCP flow
Chat, Web, Telnet, email (usr/srv)	TCP flow and inactive time > 4 s
FTP	TCP flow and inactive time > 4 s on both the control and the data channel
Games	UDP flow and inactive time > 4 s
DNS, ICMP	One request/response exchange
Audio and video	TCP flow (control channel) and a corresponding UDP flow (media traffic)

Table 2: Transaction identification

We identify requests and responses using the data flow between senders and receivers. Let A be a client that initiates some conversation with a server B. A *request* is identified as all data packets sent from A to B, before any data packet is received from B. A *reply* is identified as all data packets sent from B to A, before any new request from A. Figure 1 illustrates request and reply identification, and measurement of partial delay and whole delay values.

Email and Usenet applications have a delay bound of 4 hours and will retry a failed transaction for a limited number of times. Because researchers frequently need to run numerous experiments to gain insights into scenarios they study, which requires short experiment duration, we must infer DoS impact metrics for Email and Usenet from short experiments. We observe that DoS impact usually stabilizes shortly after the onset of an attack or after the defense's activation, unless the attack or the defense exhibit time-varying behavior. We can thus use the *pft* value measured for transactions that originate after the stabilization point as a predictor of *pft* in a longer experiment. Let r be a total number of retries within 4 hours and let s be the stabilized *pft* for email (or Usenet) transactions during a short experiment. The predicted *pft* for a long experiment is then: $pft_p = s^r$.

Finally, FTP and email success criteria require comparing a transaction duration during an attack with its expected duration without the attack. Since transaction duration depends on the volume of data being transferred and network load, we cannot set an absolute duration threshold. If we had perfectly repeatable experiments, we could guarantee that legitimate traffic transactions occur in the fixed order, with fixed arrival times and durations. We could then measure the expected transaction duration directly, running the experiment without the attack. However, some traffic generators have built-in randomness that prevents repeatable experiments. That is, they generate a specified mix of traffic but repeated runs result in different numbers, orders, interarrival times and durations of transactions. In this case, we must estimate the expected transaction duration, using information about the throughput of transactions from the same application category that complete prior to the attack. Assume a transaction T has completed in t_r seconds, sending B bytes of data, and its duration overlaps with an attack. Let *Th* be the average throughput of transactions generated by the same application as transaction T, completed prior to the attack's start. We calculate the expected duration for the transaction T as $t_e = B/Th$. If $t_r > 3 \cdot t_e$ (see Table 1), the transaction will be labeled as failed.

3.4 DoS Metrics

We aggregate the above measures of transaction success and failure into several intuitive composite metrics and describe how they capture DoS impact on network services. Section 5 provides experimental evidence that they effectively summarize the DoS effect on the network.

The ***DoS-hist*** measure shows the histogram of *pft* measures across application categories. We found this measure especially useful for capturing the impact of attacks that target only one application, e.g., TCP SYN attack at Web server port 80. Because many DoS attacks inflict damage only while they are active, and the impact ceases when the attack is aborted, we suggest that only transactions that overlap the attack be used for *DoS-hist* calculation.

The ***DoS-level*** measure is the weighted average of *pft* measures for all applications of interest: $\textit{DoS-level} = \sum_k \textit{pft}(k) \cdot w_k$, where k spans all application categories, and w_k is a weight associated with a category k. We propose this measure because in some experiments it may be useful to produce a single number that describes the DoS impact, but we caution that *DoS-level* is highly dependent on the chosen set of application weights. Unless there is a broad consensus on the appropriate set of weights, using *DoS-level* for defense performance comparison could lead to false conclusions, as weights can be chosen to bias the results in any desired way.

The ***QoS-degrade*** measure for each failed transaction is a measure of the severity of service denial. We compute this measure by locating a transaction's measurement that exceeded its QoS threshold and calculating the ratio of their difference and the threshold. For example, if d is the measured delay that exceeds the threshold value t, $\textit{QoS-degrade}=(d - t)/t$. If more than one measurement violates its threshold, we choose the largest *QoS-degrade*. Intuitively, a value N of *QoS-degrade* means that the service of failed transactions was N times worse than a user could tolerate. In experiments, we report the average of the *QoS-degrade* measures for transactions in the same application category.

The ***life diagram*** shows the birth and death of each transaction during the experiment with colored horizontal bars. The x-axis represents the time and the bar's position indicates a transaction's birth (start of the bar) and death (end of the bar). We show the failed and the successful transactions on separate diagrams, for visibility reasons. We believe that this diagram can help researchers quickly evaluate which transactions failed and spot commonalities (e.g., all are long FTP transactions, or all failed transactions are close to the start of the attack).

The ***failure ratio*** shows the percentage of transactions that are alive in the current interval (we use 1-second intervals in our experiments), but will fail (violate one of their QoS requirements) in the future. The failure ratio is especially useful for evaluation of DoS defenses, where experimenters need to calculate DoS impact over time, to capture the timeliness of a defense's response. It is also useful to capture the impact of time-varying attacks, such as pulsing floods [18]. We identify live transactions by observing each transaction whose duration overlaps or follows the attack — transactions that complete prior to the attack are excluded since they cannot be affected by it. Transactions that are born during or after the attack are considered live until they either complete successfully or fail. Transactions that are born before the attack are considered live after the attack starts. We make this adjustment of the birth time to avoid the measurement being biased by the traffic mix. Without this adjustment, we would have a positive and variable failure ratio well before the attack, which would only depend on the ratio of the lengthy transactions in the traffic mix.

A transaction that fails contributes to the failed transaction count in all intervals where it was live. We have considered an alternative approach where a transaction counts as failed only in the interval when the failure occurs. However, since it takes at least several seconds for an application to breach some threshold and fail, the failed transaction count would always lag behind the live transaction count, and failure ratio would not reach 100% in any scenario.

4. METHODOLOGY

In this section, we describe the topology and traffic scenarios in the DETER testbed [4] that we employ to illustrate our metrics. The DETER testbed at the USC Information Sciences Institute and UC Berkeley allows security researchers to replicate threats of interest in a secure environment and to develop, deploy and evaluate potential solutions. The testbed has a variety of hardware devices and supports many popular operating systems. Researchers obtain exclusive use of a portion of the testbed, configured into a user-specified topology.

4.1 Topology

The experimental topology is shown in Figure 2. It consists of four client networks and two attack networks interconnected via four core routers. Each client network has four server nodes and two client nodes. All but one client network (Net2) have an access router that connects the network to the core. We introduced this asymmetry so we could study the impact of the traffic path crossing different numbers of routers on the service quality.

All but four links in the topology have no traffic shaping. They have the default bandwidth of 1 Gbps and a delay of 0 ms[1]. Four links that connect client networks to the core have limited bandwidth of 10 Mbps and different delays of several tens of milliseconds (shown in Figure 2). We imposed the bandwidth limits to create bottleneck links that can be saturated by flooding bandwidth attacks, and we imposed different delays to break the synchronization between TCP connections. The location of bottlenecks is chosen to mimic high-bandwidth local networks that connect over a limited access link to an overprovisioned core. The limited bandwidth and delay are introduced by deploying a Click router [17] (small, black circles in Figure 2) on a bottleneck link, and running a script that emulates the desired link properties.

We simulate six application types: Web, DNS, FTP, Telnet, IRC and VoIP. In networks 1 and 2, Web, DNS, FTP and Telnet service each have their dedicated server. In networks 3 and 4, DNS and Web service are collocated on one server, FTP and Telnet service reside on another server, and IRC and VoIP services have dedicated servers. Service separation enables us to isolate effects of DoS attacks on a specific service, and attribute the impact on other traffic to network congestion. Collocated services enable us to measure the extent to which resource sharing can transfer the denial of service from a targeted service to a collocated service. Each attack network hosts two attackers.

[1]This simply means that there is no added delay. Because traffic on a link still has to cross this link and a switch there is some delay, but it is much smaller than delays we introduce through shaping.

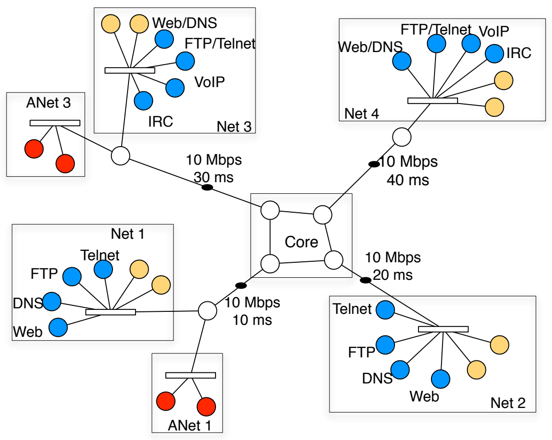


Figure 2: Experimental topology

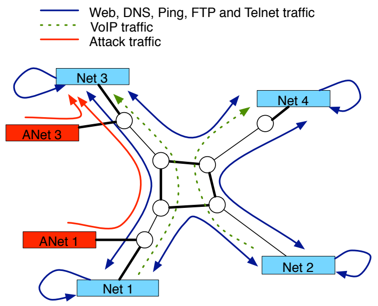


Figure 3: Communication patterns

4.2 Background Traffic

Each client generates a mixture of Web, DNS, FTP, Telnet and IRC traffic. Clients talk with servers in their own network, and with servers from two out of three external networks (see Figure 3). Specifically, clients from networks 1 and 4 talk to servers in networks 2 and 3, and clients from networks 2 and 3 talk to servers in networks 1 and 4. Additionally, one client in network 1 talks to the VoIP server in network 3, and similarly one client in network 2 talks to the VoIP server in network 4. The difference between the VoIP and other service's traffic patterns occurs due to limitations of our VoIP traffic generator. This generator can only support a conversation between a single client and a single server.

We select attack targets in network 3. Thus, client traffic from networks 1 and 3 shares the path with the attack, regardless of the destination, while client traffic from networks 2 and 4 shares the path only if its destination is in network 3. Since only clients from network 1 and 4, but not from 2 will talk with servers in network 3, we can measure various incarnations of service denial. Traffic from network 4 to network 3 will suffer the direct effect because it is destined for the target network. Traffic from network 1 to 3 will suffer both because of the congestion on the shared attack path and because it is destined for the target network. Traffic from network 2 to 4 should be free of denial of service since it neither shares a path nor travels to the attack's target. Traffic from network 2 to 1 will suffer because it shares the path with attack traffic.

We use diverse background traffic to evaluate the impact of DoS attacks on the traffic. Whenever possible, we used the real server and client applications to generate traffic, so we could faithfully replicate traffic dynamics and avoid artifacts introduced by traffic generators. File sizes, user request arrivals and transaction durations are drawn from the distributions observed in real-world traffic [19]. We had to adjust the distribution parameters to create a sufficient number of transactions within the experiment duration (10 minutes), while avoiding congestion. Hence, our request arrivals are shorter than in real world traffic (to maximize number of transactions), and our file sizes are smaller (to avoid congestion). Traffic parameters and their distributions are given in Table 4.2.

Type	Parameter (unit)	Distribution
Telnet	Time between key strokes (s)	Exp(1), S=1, M=10
	Request interarrival time (s)	Exp(5), S=5, M=100
	Response size (B)	Γ(40,0.9)
	Session duration (s)	Γ(30,0.5)
	Time between sessions (s)	Exp(2,1,10)
FTP	Request interarrival time (s)	Exp(5), S=1, M=100
	File size (B)	Pareto(1.2,5000), M=5000
HTTP	Request interarrival time (s)	Exp(5), S=1, M=15
	File size (B)	Pareto(1.04,1000), M=1000
DNS	Request interarrival time (s)	Exp(1), S=3, M=30
Ping	Request interarrival time (s)	Exp(5), S=2, M=30
IRC	Request interarrival time (s)	Exp(1), S=5, M=100
	Message size (B)	Γ(40,0.9)
VoIP	Packet interarrival time (s)	0.03

Table 3: Legitimate traffic parameters and their distributions. S is the scaling factor that multiplies the random variable drawn from the distribution. M is the maximum allowed value for the given parameter— values larger than M are scaled down to M.

We generate the following traffic types: (1) Telnet traffic is generated using interactive SSH. We generate characters that a user would type on the client machine using a perl bot and pipe them to the server via an SSH channel. The server then returns the reply via the same channel. After the current session finishes, the SSH channel is terminated (TCP connection is closed) and later reopened (new TCP connection is established) for a new session. (2) FTP traffic is generated by a client sending requests for files via wget to the server running vsFTP. Each request asks for one file only, and opens a new TCP connection. (3) HTTP traffic is generated by a client sending requests for files via wget to the server running Apache. Each request asks for one file only, and opens a new TCP connection. (4) Ping traffic is generated by a client sending one ping request to the server. (5) DNS traffic is generated by a client using dig to query a chosen server. The server runs bind which generates replies. (6) IRC traffic is generated by a client using an automated perl bot to generate chat messages to a server running pircd. (7) We generate simulated VoIP traffic that sends 128B packets every 30 ms from a client to a server.

5. EVALUATION OF METRICS

In this section, we generate several popular variants of DoS attacks and apply our metrics to evaluate attack impact on network services. While there are numerous ways to deny service in addition to the ones we tested, our chosen attacks form an exhaustive collection of attacks used for testing DDoS defenses by other researchers [20, 27, 16, 24, 30]. They are also the most commonly seen attack variants in the real DDoS incidents. Each experiment lasts for 10 minutes.

Our tests are meant only to illustrate the expressiveness of the metrics. More comprehensive tests would be needed to evaluate resilience of various services to different DoS attacks. In each

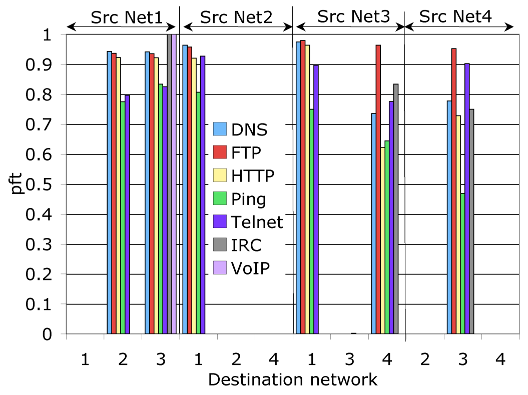


Figure 4: *DoS-hist* measures for all source and destination networks, for the UDP bandwidth flood.

subsection, we illustrate how a legacy metric (as discussed in Section 2) fails to capture the impact of DoS as expressively and completely as our proposed *pft* metrics.

5.1 UDP Bandwidth Flood

UDP flood attacks can deny service in two ways: (1) by generating a huge volume of traffic that exhausts bandwidth on the bottleneck links, (2) by generating a high packet rate that exhausts the CPU at an intermediate router or the target host. In this experiment, we generate a UDP bandwidth flood with 1000-byte packets to achieve high-bandwidth attacks with moderate packet per second rate. Four attackers located in ANet1 and ANet3 target the DNS/Web server in Net3 and generate attack packets at the maximum possible speed. The attack starts at 100 seconds and lasts for 460 seconds. The expected effect is that links connecting networks Net1 and Net3 to the core will become congested and all network traffic targeted to Net3 will experience high latencies and packets drops. Additionally, cross traffic traversing the congested links, such as traffic exchanged with Net1 or originating from Net3 will also be affected.

Figure 4 shows the *DoS-hist* measures for all source and destination networks. We calculate each *DoS-hist* measure by applying filters to the client tcpdump trace to select only the traffic for the specified destination network. Then, we run our measurement tool on this trace, and average the *pft* measures for both clients in the source network. As we expected, the *pft* measures for clients that share the same network are very similar, since they communicate with the same destinations and their traffic crosses the same links. Labels on the top of the graph indicate *DoS-hist* measures that belong to the same source network, x-axis labels denote the destination network, and the y-axis shows the *pft* per application.

Unlike our prognosis, the traffic from and to Net1 experiences the largest service denial because the attack from network ANet1 shares the bottleneck link with this legitimate traffic, and the attack completely saturates this link. Almost all traffic from and to Net1 is denied service. The slight difference between *pft* measures for different applications associated with Net1 does not result from their higher or lower resiliency to packet drops; since the generated attack is very strong, all the service should be denied. Rather, this is a side effect of our short experiment and different transaction density in two brief intervals at the beginning and the end of the attack, when queues are not too full. Transactions that "luck out" and fall in those intervals have a better chance of success, and since the test is short, even a small number of such successes can mean a

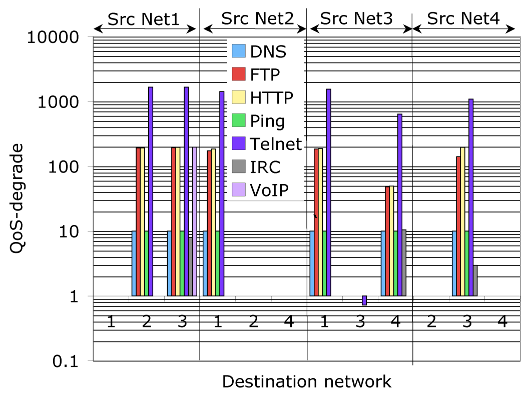


Figure 5: ***QoS-degrade*** **measure for all source and destination networks, and all application types, for the UDP bandwidth flood.**

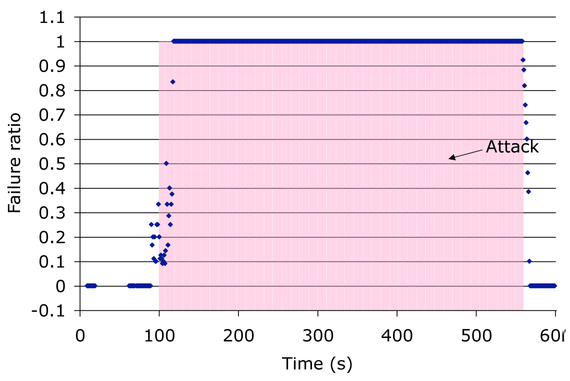


Figure 6: The failure ratio for traffic from Net1 to Net3, for UDP bandwidth flood.

significant difference in percentage.

Traffic from and to Net3 experiences lower service denial for two reasons: (1) the attack traffic from ANet3 does not traverse the bottleneck link that connects Net3 to the core and (2) the attack traffic from ANet1 arrives to the bottleneck link at a small volume (10 Mbps), because it crossed the bottleneck link in front of Net1.

These two phenomena — the influence of transaction density and experiment duration on *pft* measures, and lower service denial at Net3 than at Net1 — are artifacts of the experimental setup. They illustrate the fact that experiments must be carefully designed and include many diverse scenarios, to ensure that measurements reflect the true nature of the event being studied instead of experimental artifacts.

Figure 5 shows the *QoS-degrade* measure. All services that include traffic to or from Net1 are severely degraded, while services to and from Net3 experience smaller service denial. A small percentage of Telnet traffic is degraded in all networks because retransmissions of dropped packets from or to Net3 create congestion that violates Telnet's small echo delay bound. For space reasons, we omit the *QoS-degrade* measure for the following experiments.

Figure 6 shows the failure ratio for all transactions originating from Net1 to Net3. The periods when queues are filling and emptying, at the attack start and end, are noticeable as intervals when the failure ratio is smaller than 1. Throughout the attack, the failure ratio value stays at 1, illustrating that all service between these two networks is denied.

Figure 7 shows the life diagrams of successful and failed transactions. The x-axis plots the start and end time of a transaction, the colored bars represent transactions, and the y-axis shows the transaction ID. This is just a number we assign to a transaction to identify it. We assign consecutive numbers to transactions of the same type to make the life diagram more visible. While most of the failures occur during the attack, a few Telnet transactions fail after the end of the attack. This is because other TCP-based transactions (Web and FTP) start recovery after the end of the attack, growing their congestion window. This excess traffic causes a few Telnet packet losses and due to a low delay bound, Telnet transactions fail even from a single packet loss. Transactions that start prior to the attack fail promptly as soon as the attack is launched.

Finally, we contrast the proposed metrics with the legacy metric of request/response delay for traffic originating from Net1 in Figure 8. The graph shows the cdf function of the request/response delay metric in the baseline case with no attack, and compares it to the case when the attack is present. The x-axis plots the request-response delay on a logarithmic scale. While the distribution under attack looks different than the distribution in the baseline case, a large fraction of the transactions experience very similar delays in both cases. In fact, as illustrated by point A on the graph, some transactions that fail have a lower request/response delay than transactions that have succeeded. The transaction highlighted by point A is an IRC transaction that fails because its roundtrip time was longer than permitted by the thresholds defined in Table 1. This illustrates that request/response metric by itself is insufficient to completely capture the DoS impact on the network.

5.2 TCP SYN Flood

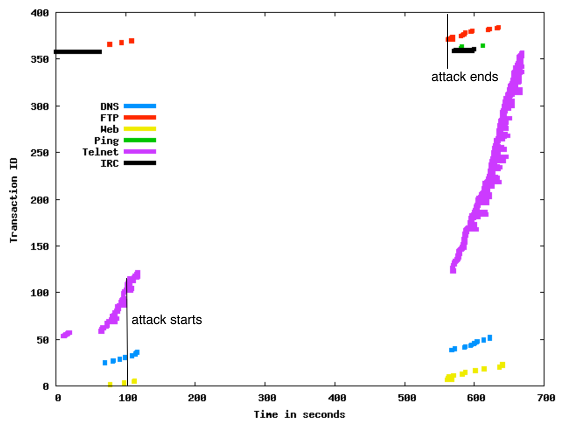
TCP SYN flood attacks deny service by generating many TCP SYN packets to an open TCP port. In this experiment, we generate 40-byte TCP SYN packets from all four attackers to the Web server in Net3. Each attack machine sends 200 packets per second, which is sufficient to deny service to Web traffic at the target machine. Since TCP SYN packets are small, no congestion should build up and no other network service should be denied. The attack starts at 100 seconds and lasts for 400 seconds.

Figure 9 shows the *DoS-hist* measures for all source and destination networks. Almost all Web transactions with Net3 have failed as a result of this attack. The *pft* of Web transactions is a little lower than 1 thanks to the transactions that start close to the end of the attack. These transactions recover after the attack stops and before their delay exceeds the threshold.

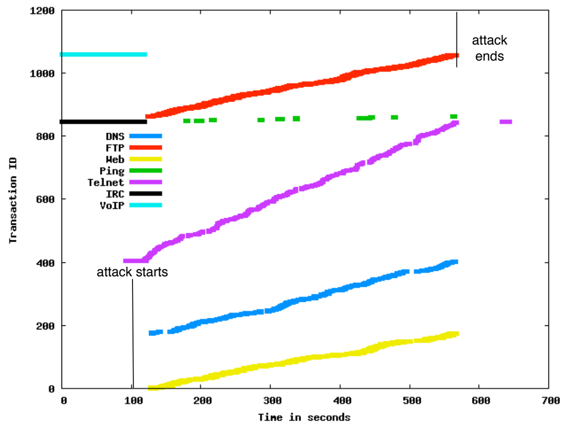
Figure 10 shows the failure ratio for Web transactions originating from Net1 to Net3. For almost the entire attack duration, the failure ratio value stays at 1, occasionally dropping to lower values for 1 second when some legitimate SYN packet gains access to the target's connection buffer.

Figure 11 shows the life diagrams of successful and failed transactions. All the failures occur during the attack and only Web transactions fail. One Telnet transaction fails also, due to a single packet loss, caused by mild congestion.

Next, we compare the *pft*-based metrics to the legacy metric of transaction duration for traffic originating from Net1 in Figure 12. The duration distributions in the attack and the baseline case appear very similar, and failed transactions have the same or lower duration than about 20% of successful transactions under attack, and 20% of baseline transactions. We have highlighted one such point B on the graph, where an HTTP transaction fails because its roundtrip time exceeded the delay bound. This illustrates that the



(a) Successful transactions



(b) Failed transactions

Figure 7: Life diagram of successful and failed transactions, for UDP bandwidth flood.

transaction duration metric by itself fails to completely capture the impact of the attack on the network traffic.

5.3 ICMP CPU Flood

Similar to UDP flood attacks, ICMP floods target bandwidth or CPU resources. In this experiment, we generate an ICMP CPU flood by directing 48 byte ICMP packets from all the attackers in ANet1 and ANet3 to the victim DNS/Web server in Net3. Each attacker generates a 10K packets per second flood; the attack starts at 110 seconds and lasts until 560 seconds. The expected effect is that routers connecting networks Net1 and Net3 to the core will become overwhelmed and drop all traffic. Thus, all traffic to Net3 should experience drops and high latencies, in addition to traffic from Net1 to Net2 and Net3.

Figure 13 shows the *DoS-hist* measures for all source and destination networks. All transactions to and from Net1 experience the largest service denial, for the same reason as in the case of the UDP bandwidth flood — the sharing of a common router by the attack from network ANet1 and the legitimate traffic to and from Net1 makes this traffic the most affected. However, we observe that the impact of ICMP CPU flood attacks is lower than the impact of UDP bandwidth floods. This is because attack packets are small, so router buffers manage to successfully store and forward more legitimate packets even though they experience higher latencies. This results in a lower percentage of lost legitimate packets. For example, 60-80% of Ping transactions, 75-90% of Web transactions and about 70% of DNS transactions with Net1 were successful in spite of the attack. Web transactions are denied the least because dropped packets will be retransmitted by TCP, increasing the chance of transaction success. FTP and Telnet traffic suffered the largest denial. FTP transactions got prolonged and violated their overall duration bound, while Telnet traffic easily exceeded its small echo delay bound even when packet loss was small. This is visible when observing the attack effect on transactions to and from Net3, where only Telnet traffic suffered around 55% failure.

Figure 14 shows the failure ratio for all transactions originating from Net1 to Net3. The failure ratio was oscillating throughout the attack, as the routers were trying to keep up with the packet flood.

Finally, we illustrate how the legacy metrics fail to capture the details of the failure conditions. We use two legacy metrics: packet loss for traffic originating from Net1, and throughput of the traffic originating from Net1. In Figure 15, the x-axis represents the fraction of packets lost in a transaction. We observe that although the loss is higher during an attack, many failed transactions have lower loss than successful ones. These transactions fail due to latency intolerance. We have highlighted one such point C on the graph, where a Telnet transaction has 8% loss but fails due to a high round-trip time. Similarly, in Figure 16, the y-axis represents the average throughput on a logarithmic scale. Even though the distributions in the baseline and the attack case appear similar, there are many failed transactions with throughput higher than that of successful transactions. We have highlighted one such point D on the graph, where a Telnet transaction fails since its roundtrip time exceeded the echo delay threshold.

5.4 HTTP Flood (Flash Crowd Attack)

Flash crowd attacks deny service by sending many legitimate-like requests to a server whose resources are limited. The primary limitation usually lies in the server application that cannot handle more than a certain number of simultaneous requests. To mimic flash crowd attacks, we have modified the Web server's configuration file in Net3 to limit the number of clients and the number of client requests that can be served simultaneously, to 50 and 100 respectively. Only one attacker from ANet1 participates in HTTP flood, sending 100 requests per second to the Net3's Web server, and requesting a small, 5-byte file. The reply size is deliberately made small, so we could guarantee that the reverse traffic will not create congestion. The attack starts at time 110 seconds and lasts

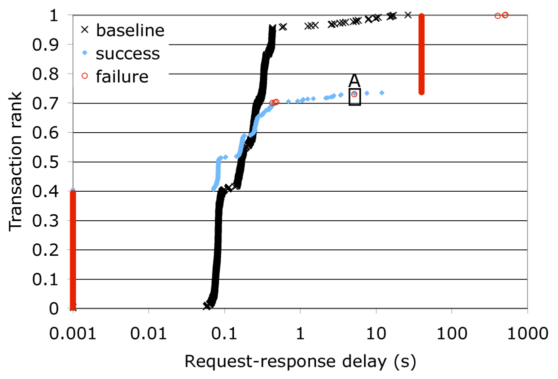


Figure 8: The distribution of request/response delay for traffic from Net1, for UDP bandwidth flood.

for 2 minutes. We expect that the flash crowd effect should be similar to the effect of TCP SYN attack, that is, legitimate Web requests to the targeted server should be denied service while other traffic should not be affected. We observe this effect in the two graphs illustrating the *DoS-hist* and the failure ratio of Web transactions.

Figure 17 shows the *DoS-hist* measures for all source and destination networks. Similar to the case of TCP SYN flood attack, almost all Web transactions with Net3 have failed. Additionally, a very small percentage (0.5-0.7%) of Telnet and FTP transactions fail because of increased congestion. Figure 18 shows the failure ratio for Web transactions only originating from Net1 to Net3. A striking difference between this and other flooding attacks is that the DoS impact remains even after the attack. Excess Web requests seem to permanently disable the Apache Web server and we could only restore it by restarting the Apache process on the server host.

5.5 Pulsing Attack

Pulsing attacks, also known as low-rate TCP attacks [18], deny service by periodically creating congestion on the path shared with legitimate TCP traffic. This leads to traffic drops and the legitimate TCP traffic responds by decreasing its sending rate. Denial of service occurs because the goodput and throughput of affected TCP connections are significantly reduced due to the congestion control response, even when the pulses are relatively wide apart. Pulsing attacks are appealing to the attackers because attack traffic can be sent stealthily in short, sparse pulses, making detection difficult. We generate a UDP pulsing attack with the same parameters as in the case of UDP bandwidth floods. The pulses start at 195 seconds, last for 20 seconds, with a sleep time between pulses of 100 seconds. There is a total of 5 pulses. Although [18] suggests that pulses as short as 1 RTT are sufficient to deny service, our pulses last longer because our traffic is sparse and we wanted to maximize the chance that an attack overlaps with the legitimate TCP traffic.

We observe that the pulsing flood denies traffic to and from Net1, and to some small extent to and from Net3. The denial is smaller than in the case of the UDP flood attack because resources are intermittently consumed.

Figure 20 shows the failure ratio for transactions originating from Net1 to Net3. The failure ratio oscillates with the attack, but the transactions fail even when the attack is not present because the periodic loss inflicts significant damage on their sending rate during loss periods and they cannot recover quickly enough when the attack stops to meet their QoS criteria. We compare the failure ratio to the legacy metric of division of resources for bandwidth reach-

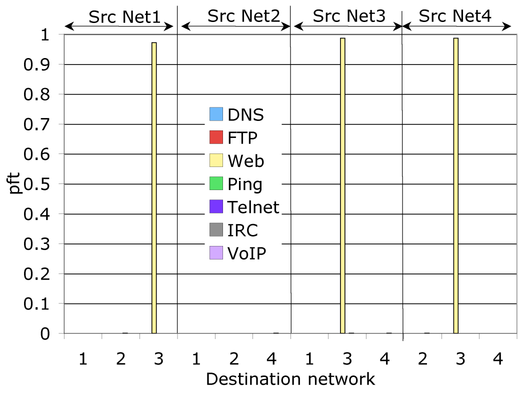


Figure 9: *DoS-hist* measure for all source and destination networks, and all application types, for TCP SYN flood attack.

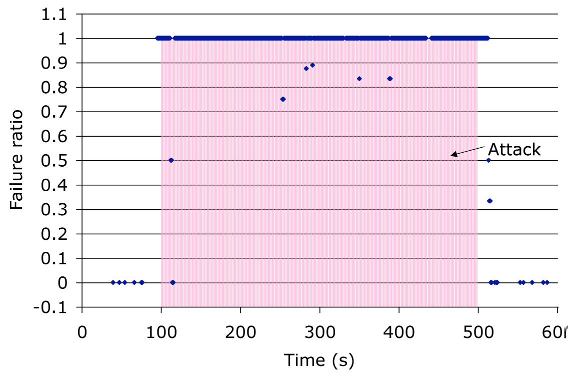
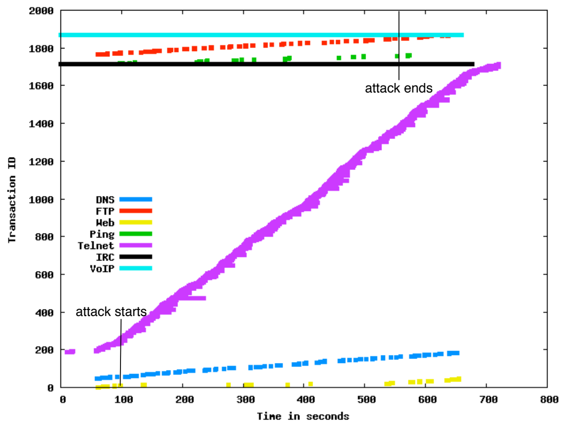
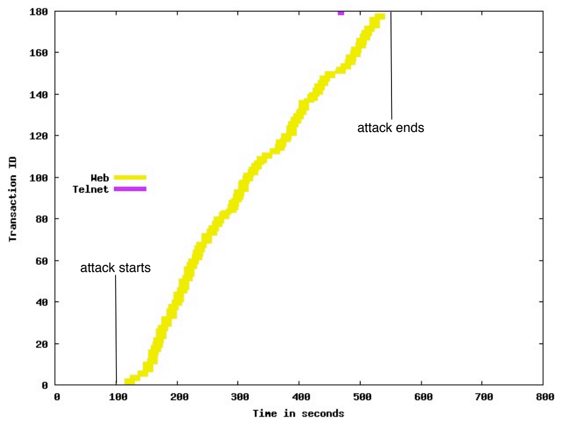


Figure 10: The failure ratio for Web transactions in traffic from Net1 to Net3, for TCP SYN flood attack.



(a) Successful transactions



(b) Failed transactions

Figure 11: Life diagram of successful and failed transactions, for TCP SYN flood attack.

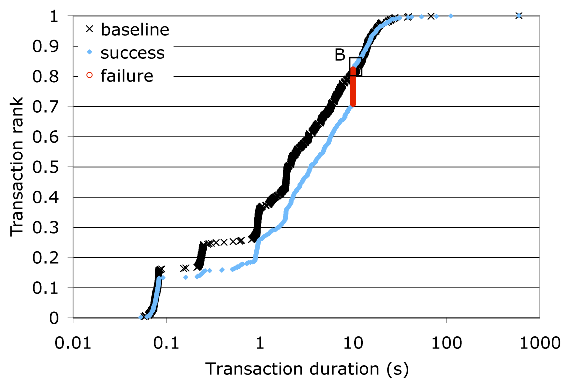


Figure 12: The distribution of transaction duration for traffic from Net1, for TCP SYN flood attack.

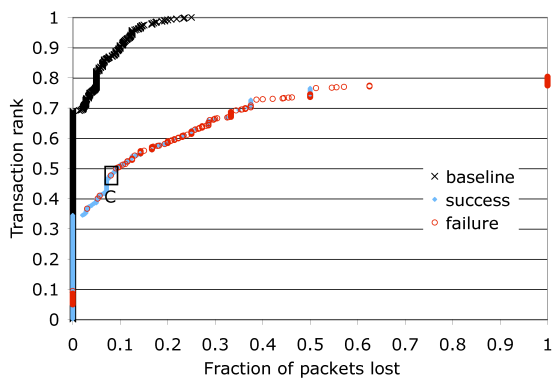


Figure 15: The distribution of packet loss for traffic from Net1, for ICMP CPU flood.

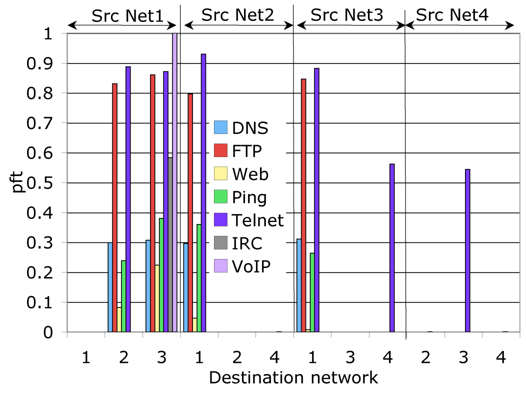


Figure 13: *DoS-hist* measure for all source and destination networks, and all application types, for ICMP CPU flood.

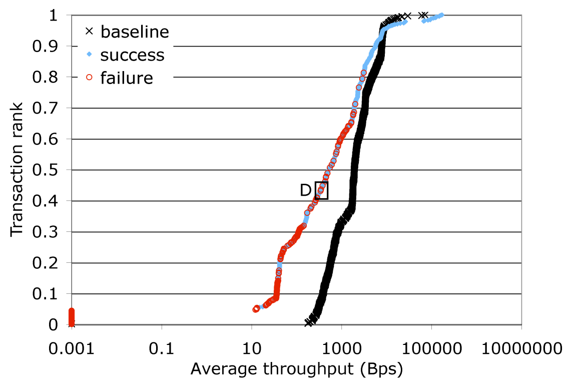


Figure 16: The distribution of throughput for traffic from Net1, for ICMP CPU flood.

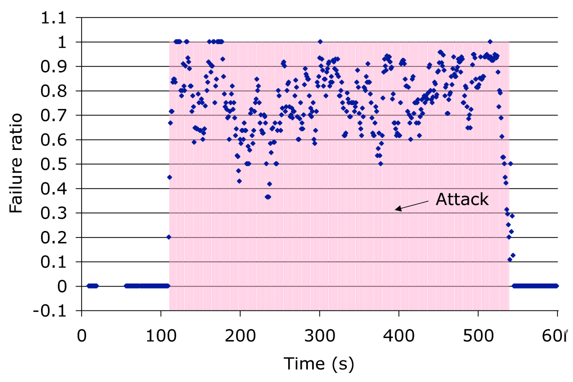


Figure 14: The failure ratio for all transactions in traffic from Net1 to Net3, for ICMP CPU flood.

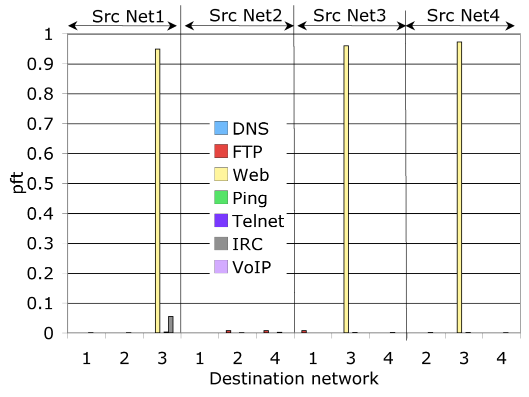


Figure 17: *DoS-hist* measure for all source and destination networks, and all application types, for HTTP flood.

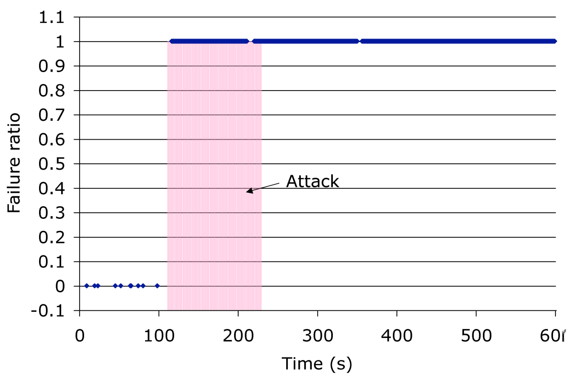


Figure 18: The failure ratio for Web transactions in traffic from Net1 to Net3, for HTTP flood.

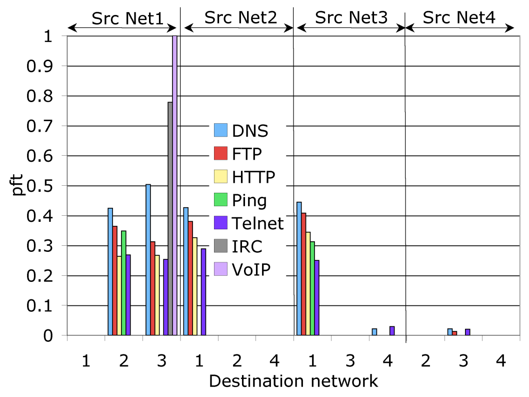


Figure 19: *DoS-hist* measures for all source and destination networks, for pulsing flood.

ing the victim server in Figure 21. The y-axis shows the percentage of bandwidth consumed by legitimate or attack traffic. Although this measure does indicate that the period of service denial is during the pulses, it does not capture the impact of the attack between the pulses. Comparing Figure 20 and Figure 21, we observe that the failure ratio metric clearly captures both the effect of the attack during the pulses and in-between them, making it a superior metric.

5.6 Defense Effect

We now illustrate how our proposed DoS impact metrics can be used to measure the effectiveness of a defense. We repeat the TCP SYN flood attack, but we turn on the TCP SYN cookies option after the first 3 minutes of the attack. SYN cookies prevent resource allocation at the server until the TCP 3-way handshake is complete, thus disabling the TCP SYN flood means of service denial. We expect that shortly after the activation of SYN cookies, the Web service quality should return back to normal.

Figure 22 shows the *DoS-hist* measures for all source and destination networks. Comparing it with Figure 9 for the TCP SYN attack, we observe that when the SYN cookie defense in activated, the *pft* becomes significantly lower than during a SYN flood without the defense. Figure 23 shows the failure ratio for Web transactions from Net1 to Net3. Initially, all the Web transactions fail. After the defense deployment, this ratio returns to 0, illustrating a complete protection from the attack.

6. NS-2 IMPLEMENTATION

To extend the application of our proposed metrics to simulated DDoS defense evaluation, we have ported the metrics to the NS-2 simulator [23]. During simulation, we generate flows such that each flow exactly represents a transaction. Since we specify a unique ID for each transaction, we can easily compute the duration, echo, partial and whole request/response delay for a transaction. We then compare measured values with QoS requirements, and produce the *DoS-hist* measure.

We illustrate the DoS impact metrics in small-scale experiments using the NS-2 (version 2.29) simulator and we compare it with identical experiments in the DETER testbed. The simple network topology contains a single legitimate client, an attacker, and a server. All nodes are connected to the same router. The link between the server and router is 10 Mbps with 10 ms delay. The other two links are 100 Mbps bandwidth with 10 ms delay. We use a queue size of 100 packets, with a drop-tail queuing strategy. We generate the following legitimate traffic between the client and the server: (1) Web and FTP traffic with file size 1000 bytes and 20 second request interarrival period. (2) Telnet traffic with 10 packets per second and a 100-byte packet size. During the simulation, we start a new Telnet connection every 60 seconds with duration 120 seconds. (3) DNS and ICMP traffic with 20 second request interarrival period. We use the following applications in NS-2 to generate the simulation traffic: Application/FTP for FTP, PagePool/WebTraf for HTTP, Application/Telnet for Telnet, Agent/Ping for ICMP, and a modified version of Agent/Ping with a maximum of 3 retransmissions with 5-second timeouts as DNS. In our simulations, we generate a UDP flood that overwhelms the bottleneck link with 10 Mbps (moderate attack) or 80 Mbps (large attack) rate.

To minimize the effect of different traffic generation dynamics in simulation and emulation, we fixed the traffic patterns so we can guarantee that both implementations observe the same transactions. The request interarrival period for Web and FTP was fixed at 20 s, the file size at 1 MB, the request interarrival period for Ping and DNS at 10 s, the Telnet session interarrival period at 1 minute, the duration at 2 minutes, and the packet rate at 10 packets per second.

Figure 24 shows the *DoS-hist* measure during the two attacks for NS-2 and DETER experiments. The x-axis shows the attack strength, and the column height denotes the result of 10 test runs. Since the legitimate traffic pattern is fixed for the NS-2 simulation, we achieve variability by randomly choosing a small delay (10-100 ms) to apply to the attack start time. We also show the DoS-level measure using equal application weights as a blue line across

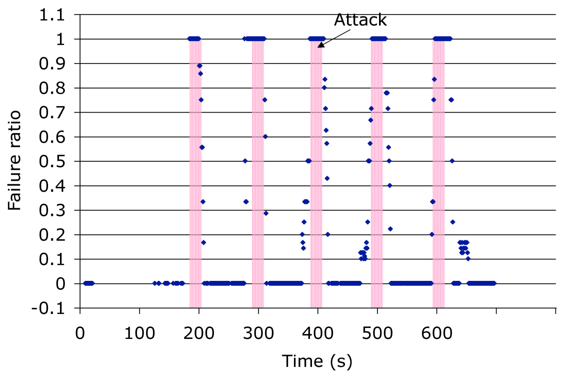


Figure 20: The failure ratio for traffic from Net1 to Net3, for pulsing flood.

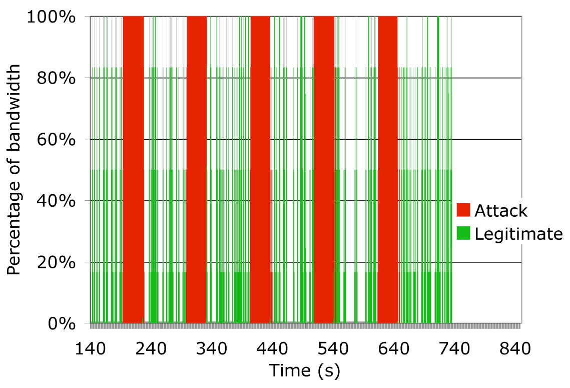


Figure 21: The division of bandwidth to Net3 during a pulsing attack

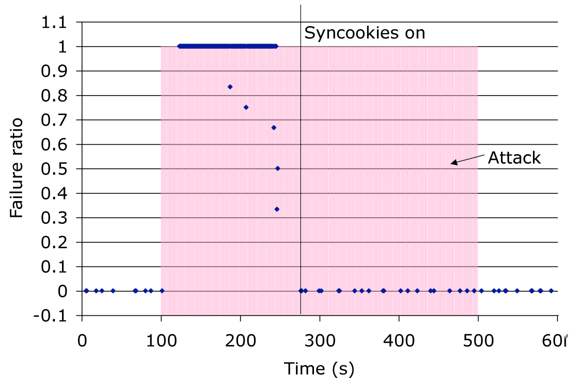


Figure 23: The failure ratio for Web transactions in traffic from Net1 to Net3, for TCP SYN flood with dynamic SYN cookie defense.

the histograms in the Figure. The Telnet application is the most affected by the attack due to its small echo-delay bound (250 ms). Denial of service is similar for DNS and Ping, even though DNS can retransmit requests up to three times because these retransmissions occur after the DNS' request/response delay threshold is exceeded. Web transactions survive the attack best because of the generous (10 s) delay threshold and because the lost requests are retransmitted by the TCP. At high attack rate (80 Mbps), the *pft* of all applications goes to almost 100%.

Comparing simulation results with testbed results, shown in Figure 24(b), we find that trends in both graphs are the same but more transactions fail in simulations. This is because the software routers used on the testbed can handle the attack traffic better, than the simple single output queuing model used in NS-2. These results are consistent with the results in [9] which show higher throughput and TCP congestion window sizes in testbed experiments compared to same experiments in NS-2.

7. RELATED WORK

In the quality of service field, there is an initiative, led by 3GPP partnership, to define a universally accepted set of QoS requirements for applications [1]. While many of the specified requirements apply to our work, we extend, modify and formalize these requirements as explained in Section 3.2.

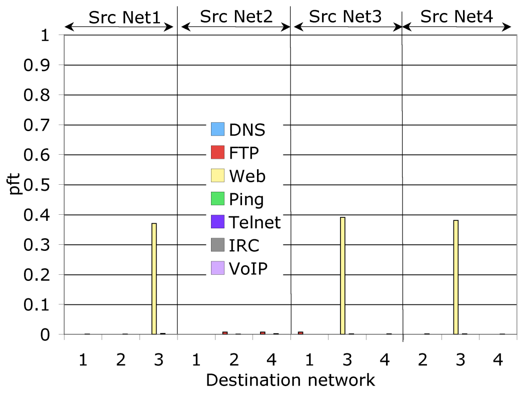
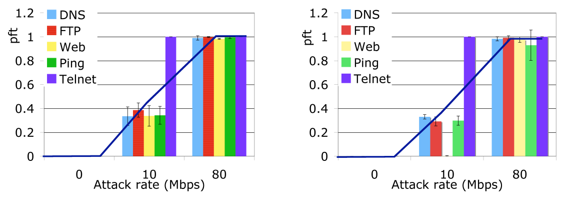


Figure 22: *DoS-hist* measure for all source and destination networks, and all application types, for TCP SYN flood with dynamic SYN cookie defense.



(a) NS-2 (b) DETER

Figure 24: *DoS-hist* and *DoS-level* measures in NS-2 and DETER experiments

The Internet and ATM research communities have separated applications into several categories based on their sensitivity to delay, loss and jitter [12]. An application is either inelastic (real-time), which requires end-to-end delay bounds, or elastic, which can wait for data to arrive. Real-time applications are further subdivided into those that are intolerant to delay, and those that are more tolerant, called "delay-adaptive." The Internet's integrated services framework mapped these application types onto three service categories: the guaranteed service, the controlled load service and the currently available best effort service [6]. These research efforts, however, focus on providing guaranteed service to applications, rather than on measuring if the service was denied during a DoS attack.

In study [7], the authors measure DoS impact on real-world traffic via the distributions of several parameters: the throughput of FTP applications, round-trip times of FTP and Web flows, and latency of Web flows and the DNS lookup service in real world traces before, during, and after an attack. Our paper strives to define a more formal threshold-based model for these and other parameters, that can be extended to a broader variety of services and attacks.

The Internet Research Task Force Transport Modeling Research Group (TMRG) is chartered to standardize evaluation of transport protocols by developing a common testing methodology, including a benchmark suite of tests [13]. The TMRG documents discuss using user-based QoS metrics for measuring congestion, but do not specify such metrics in any detail.

In [8] a user satisfaction index is computed from Skype traces using regression analyses of several call and quality of service attributes. The index is validated via analysis of other call characteristics, such as conversation interactivity. Our work provides a framework where this index can be easily incorporated into a DoS metric for Skype and other VoIP traffic.

In slow-motion benchmarking [22], the authors modify desktop benchmarks by introducing delays between tasks, and use network traces collected at the client to measure performance of thin clients. The only performance measure in that work is the sum of transaction durations in the benchmark.

8. CONCLUSIONS AND FUTURE WORK

Ultimately, DoS attacks are about denying end user service. A complete DoS metric has to intuitively and succinctly summarize end user conditions that truly capture the service denial aspect of the attack. We believe the key aspect of designing such a metric is defining a threshold-based model to capture the quality of service expectations of the end user. The *p*ft metric proposed in this paper elegantly captures the impact of the attack as experienced by the end user. We are able to represent the denial of service effects with high accuracy for a wide range of attacks and have illustrated inadequacy of legacy metrics to capture these effects.

We believe there is much more work to be done in defining an appropriate threshold-based model that encapsulates all aspects of network and attack traffic. However, the techniques outlined in this paper provide a strong framework to the research community to develop a formal methodology and metric for malicious traffic seen on the network. This is the first concentrated effort in developing unbiased metrics for DoS technology evaluation and will hopefully encourage such endeavors in the future.

9. REFERENCES

[1] 3GPP. *The 3rd Generation Partnership Project (3GPP).*

[2] J. Ash, M. Dolly, C. Dvorak, A. Morton, P. Taraporte, and Y. E. Mghazli. Y.1541-QOSM – Y.1541 QoS Model for Networks Using Y.1541 QoS Classes. NSIS Working Group, Internet Draft, Work in progress, May 2006.

[3] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *In Proceedings of ACM NetGames 2004*.

[4] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experiences With DETER: A Testbed for Security Research. In *2nd IEEE TridentCom Conference*, March 2006.

[5] Nina Bhatti, Anna Bouch, and Allan Kuchinsky. Quality is in the Eye of the Beholder: Meeting Users' Requirements for Internet Quality of Service. Technical Report HPL-2000-4, Hewlett Packard, 2000.

[6] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633, June 1994. `http://www.ietf.org/rfc/rfc1633.txt`.

[7] Kun chan Lan, Alefiya Hussain, and Debojyoti Dutta. The Effect of Malicious Traffic on the Network. In *Passive and Active Measurement Workshop (PAM)*, April 2003.

[8] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. Quantifying Skype User Satisfaction. In *Proceedings of the ACM SIGCOMM*, September 2006.

[9] R. Chertov, S. Fahmy, and N. Shroff. Emulation versus Simulation: A Case Study of TCP-Targeted Denial of Service Attacks. In *Proceedings of the 2nd International IEEE CreateNet TridentCom Conference*, February 2006.

[10] B. N. Chun and D. E. Culler. User-centric Performance Analysis of Market-based Cluster Batch Schedulers. In *In Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, May 2002.

[11] Cooperative Association for Internet Data Analysis. CAIDA Web page. `http://www.caida.org`.

[12] M. W. Garrett. Service architecture for ATM: from applications to scheduling. *IEEE Network*, 10(3):6–14, May/June 1996.

[13] IRTF TMRG group. The Transport Modeling Research Group's Web Page. `http://www.icir.org/tmrg/`.

[14] M. Guirguis, A. Bestavros, and I. Matta. Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources. In *Proceedings of ICNP*, Oct 2004.

[15] Hani Jamjoom and Kang Shin. Persistent Dropping: A Efficient Control of Traffic Aggregates. In *ACM SIGCOMM Conference*, 2003.

[16] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Berger. Botz-4-Sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds. In *NSDI*, 2005.

[17] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[18] A. Kuzmanovic and E. W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants). In *Proc. of ACM SIGCOMM*, August 2003.

[19] S. Luo and G. Marin. Modeling Networking Protocols to Test Intrusion Detection Systems. *Proceedings of the LCN'04*, 2004.

[20] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. In *ACM Computer Communication Review*, July 2001.

[21] Nortel Networks. *QoS Performance requirements for UMTS*. The 3rd Generation Partnership Project (3GPP). `http://www.3gpp.org/ftp/tsg_sa/WG1_Serv/TSGS1_03-HCourt/Docs/Docs/s1-99362.pdf`.

[22] Jason Nieh, S. Jae Yang, and Naomi Novik. Measuring Thin-Client Performance Using Slow-Motion Benchmarking. *ACM Transactions on Computer Systems*, 21(1), February 2003.

[23] The Network Simulator ns 2. NS-2 Web page. `http://www.isi.edu/nsnam/ns/`.

[24] G. Oikonomou, J. Mirkovic, P. Reiher, and M. Robinson. A Framework for Collaborative DDoS Defense. In *Proceedings of ACSAC*, December 2006.

[25] WIDE Project. MAWI Working Group Traffic Archive. `http://tracer.csl.sony.co.jp/mawi/`.

[26] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. The Effect of Latency on User Performance in Warcraft III. In *In Proceedings of ACM NetGames 2003*.

[27] Angelos Stavrou, Angelos D. Keromytis, Jason Nieh, Vishal Misra, and Dan Rubenstein. MOVE: An End-to-End Solution to Network Denial of Service. In *NDSS*, 2005.

[28] A. Yaar, A. Perrig, and D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Security and Privacy Symposium*, 2004.

[29] L. Yamamoto and J. G. Beerends. Impact of network performance parameters on the end-to-end perceived speech quality. In *In Proceedings of EXPERT ATM Traffic Symposium*, September 1997.

[30] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting Network Architecture. In *ACM SIGCOMM Conference*, 2005.